Some Issues on Control of Discrete Event Systems Using Model Specifications

Spathopoulos, M.P.; Smedinga, R.

Published in:
Proceedings on the international workshop on Discrete Event Systems, WODES96

SOME ISSUES ON CONTROL OF DISCRETE EVENT SYSTEMS USING MODEL SPECIFICATIONS

M.P. Spathopoulos[1] R. Smedinga[2]

[1] Division of Dynamics and Control, University of Strathclyde, 75 Montrose St., Glasgow, G1 1XJ Scotland, UK, email: mps@mecheng.strath.ac.uk
[2] Department of Computing Science, University of Groningen, Groningen, The Netherlands, email: rein@cs.rug.nl

Keywords
Logical discrete event system, Specification, Control Objective, Predicate, Partial Observation, Controllable

Abstract

In this paper the control of a logical discrete-event system is introduced using predicates and associated blocking events. The blocking occurs when a predicate concerning the system behaviour becomes true. It is shown that model specifications can be transformed into control objectives involving predicates and blocking events. Next, we consider the control problem when events are unobservable and uncontrollable. It is shown that when the control objectives are decidable a separation principle holds, i.e., the problems of uncontrollability and unobservability can be solved separately and independently.

1 Introduction

Different ways exists to define a logical discrete event system. A simple but effective way is the following:

$$P = \langle \mathbf{a}P, \mathbf{b}P \rangle$$

This denotes a DES with possible events collected in the finite set $\mathbf{a}P$ and with possible behaviour collected in the set $\mathbf{b}P \subseteq (\mathbf{a}P)^*$. $\mathbf{b}P$ is a set of strings. For simplicity we suppose that $\mathbf{b}P$ is prefix closed (i.e., if $st \in \mathbf{b}P$ then also $s \in \mathbf{b}P$, where st denotes concatenation of s and t). DESs defined in this way can be displayed using state graphs. The number of states in such a graph is finite if the language defined by $\mathbf{b}P$ is regular.

Controlling a DES can be done in different ways. One way is to introduce a second system, called the supervisor, that follows the system and blocks events, if necessary, in order to get desired behaviour given in terms of a finite automaton. This is the Ramadge/Wonham approach, see [1, 2]. Alternatively, the second system can be a controller in the sense that plant and controller lead to some form of desired behaviour in synchronized cooperation, see [3].

Intuitively, we can think of a DES as a black box, observed by an observer from a distance. The observer is unaware of the model of the system he is observing, he only sees events happen. The observer writes down on a piece of paper each event that occurs. Because he is unable to write down more than one event symbol at a time, events occurring in parallel will be written down on paper in some (arbitrary) order. This observation leads to a growing string of symbols on that piece of paper, representing possible behaviour of the system.

Our view of controlling simply means inspecting that written string, each time a new event happens (e.g., is written down) and, depending on some condition, block some events. It is shown that any model specification can be transformed into the so-called control objectives. Although the technique developed in this paper holds for arbitrary observed behaviour, it should be clear by now that the observer has to check only one specific string of events, namely the string of events that actually occurs.

Thus, the controller is a collection of control objectives which are evaluated on-line for the current string only (pathwise). Depending on the evaluated conditions on this string only, some events are blocked. If an event is observed next, all blocking is reset, conditions are re-evaluated and a (possibly different) set of events is blocked, see [4].

The controller involves two levels of hierarchy. In the upper level the current string is stored. In the lower level the control objectives are evaluated.
When some control objectives are uncontrollable a back-propagated algorithm is given to derive the controllable objectives. Under some assumptions this algorithm is efficient. The derivation of the observable control objectives is also given and it is proved that, when these objectives are decidable, the control problem can be solved under unobservable and uncontrollable conditions separately and independently.

proc. of the workshop on Discrete Event Systems
This work is partly supported by the Dutch Foundation for Scientific Research (NWO) and the British Council.

2 Notation

With ab we denote concatenation of events a and b (first a, then b), with $a \mid b$ choice (a or b), with a^* repetition (Kleene star operator), and with $a \parallel b$ interleaving (a and b occur in parallel). Because of the fact that events are considered to occur in infinitesimal time, $a \parallel b$ equals $ab \mid ba$. $\lceil$ denotes alphabet projection (restriction):

$$\begin{array}{ll} (xa)\lceil a = x\lceil a & \\ (xb)\lceil a = (x\lceil a)b & \text{if } b \neq a \end{array}$$

By $\mathbf{pref}(x)$ we denote the prefix closure of the string x. Similar $\mathbf{pref}(T)$ denotes the prefix closure of the language T. If s is a string of events, say $s = xb$ (with x another string and b an event) then $\mathbf{last}(s) = b$ and $\mathbf{past}(s) = x$ denote the last event of the behaviour s and the past of the behaviour respectively.

Definition 1 *A set of control objectives for a system $P = \langle \mathbf{a}P, \mathbf{b}P \rangle$ is defined by*

$$\begin{array}{ll} \Theta = & \{[B, A] \\ & : B\colon (\mathbf{a}P)^* \to \{\mathbf{true}, \mathbf{false}\} \ \wedge \ A \subseteq \mathbf{a}P \cup \{\epsilon\} \\ & \} \end{array}$$

B is the blocking condition *and A the corresponding* blocking event set.

A control objective $[B, A]$ blocks the events in A if, for a behaviour $s \in \mathbf{b}P$, the condition $B(s)$ returns true.

Definition 2 *A system $P = \langle \mathbf{a}P, \mathbf{b}P \rangle$ under the set of control objectives $\Theta = \{[B, A]\}$ is defined to be the system P_Θ given by:*

$$P_\Theta = \begin{cases} \langle \mathbf{a}P, \emptyset \rangle & \text{if } (\exists [B, A] \in \Theta : \epsilon \in A \ \wedge \ B(\epsilon)) \\ \langle \mathbf{a}P, \{t : t \in \mathbf{b}P \ \wedge \ t \, \mathbf{sat} \, \Theta\} \rangle & \text{otherwise} \end{cases}$$

where ϵ is the trigger event and

$$\begin{array}{rcl} (sbu) \, \mathbf{sat}[B, A] & = & B(s) \ \wedge \ b \notin A \\ t \, \mathbf{sat} \, \Theta & = & (\forall [B, A] \in \Theta : t \, \mathbf{sat} [B, A]) \end{array}$$

$\mathbf{b}P_\Theta$ is called the legal behaviour *of P.*

It is clear that

$$\begin{array}{rcl} \{[B_1, A], [B_2, A]\} & = & \{[B_1 \ \wedge \ B_2, A]\} \\ \{[B, A_1], [B, A_2]\} & = & \{[B, A_1 \cup A_2]\} \end{array}$$

In the remainder of this paper we assume that control objectives contain only one blocking event. If not, equations like the ones above can be used to achieve this. We will write $[B, a]$ instead of $[B, \{a\}]$. Instead of the function name B we sometimes write the corresponding expression, i.e., write $[B(s), A]$.

Example 1 Suppose the system P is given as $\langle \{a, b, c\}, (a|b|c)^* \rangle$. It is clear that P is regular, i.e., can be represented by a finite state graph. Moreover, suppose, we want to control the system in such a way that first some a's occur, next event b occurs, and last the same many c's as a's occur, i.e., $a^n b c^n$ (for arbitrary n). Clearly, the desired behaviour is no longer regular. Nevertheless, three control objectives are sufficient in order to describe this desired behaviour:

$$\begin{array}{ll} \Theta = & \{[s \, \mathbf{N} \, b = 1, \{a, b\}] \\ & , [s \, \mathbf{N} \, b = 0, \{c\}] \\ & , [s \, \mathbf{N} \, a = s \, \mathbf{N} \, c, \{a, b, c\}] \\ & \} \end{array}$$

where $s \, \mathbf{N} \, e$ denotes the number of occurrences of event e in trace s (with $\epsilon \, \mathbf{N} \, e = 0$ for each event e). It is clear that, in the controlled system, the event c is blocked as long as no b has happened and a and b are blocked as soon as b has happened. Finally, all events are blocked as soon as the desired behaviour has occurred. So we have:

$$P_\Theta = \langle \mathbf{a}P, \mathbf{pref}\{a^n b c^n : n \geq 0\} \rangle$$

3 Transforming model specifications

Practical experience shows that usually is possible to define desired behaviours in terms of control objectives. Moreover, desired behaviours are often stated using such objectives: "if this happens, block those events." Instead of translating these objectives to languages (or state graphs) our proposed method uses them directly.

3.1 General transformation

The control requirements can be considered as boolean conditions $R(s)$ on the observed behaviour s. For example, the mutual exclusion of two events a and b reads as: $R(s) =$ "in s pairing of ab and ba is not allowed to occur". This condition can be transformed into a control objective. In fact, we have

Theorem 1 *If $R\colon (\mathbf{a}P)^* \to \{\mathbf{true}, \mathbf{false}\}$ is a predicate on the behaviour, it can be replaced by the set of control objectives:*

$$\Theta = \{[B_e, e] : e \in \mathbf{a}P\}$$

where B_e: $(\mathbf{a}P)^* \to \{\mathbf{true}, \mathbf{false}\}$ *is given by:*

$$B_e(s) = R(s) \ \wedge \ \neg R(se)$$

In words the theorem says: for each event $e \in \mathbf{a}P$: block this event e if both the current observation makes the requirement R true and, when e occurs next, the requirement becomes false. In fact: we replace the predicate by a set of control objectives, namely one control objective for each event $e \in \mathbf{a}P$. The first condition of $B_e(s)$ evaluates to true by requirement and recursion ($R(s)$ cannot return false because of the blocking in the previous step) and need not be included.

An objective does not contribute to the control of the system if the blocking condition $B_e(s)$ for an event e returns false for any observation s. Sometimes we may decide whether such a condition never returns true, for example if the condition involves the number of occurrences of some events not including e. In general, the issue is undecidable. However, the inclusion of control objectives which are not needed, affect only the computations, making the design more complicated.

We recall here, that the new predicates B_e are computed off-line, while the control is performed online using the observed behaviour only.

3.2 Some examples

In this section we give some examples of control objectives.

mutual exclusion of events a and b: (if a occurs b should not occur next and visa versa)

$$\Theta = \{[\mathbf{last}(s) = a, b], [\mathbf{last}(s) = b, a]\}$$

mutual participation of a and b: (if a occurs, b occurs next, and visa versa).

$$\begin{array}{ll}\Theta = & \{\ [\mathbf{last}(s) = a \ \wedge \ \mathbf{last}(\mathbf{past}(s)) \neq b \\ & \quad , \mathbf{a}P \setminus \{b\}] \\ & , \ [\mathbf{last}(s) = b \ \wedge \ \mathbf{last}(\mathbf{past}(s)) \neq b \\ & \quad , \mathbf{a}P \setminus \{a\}] \\ & \}\end{array}$$

Example 1 (cont.) The desired behaviour can be formally expressed by the specification:

$$R(s) = (s \in \mathbf{pref}\{a^n b c^n\})$$

We use theorem 1 to show that this specification can be transformed into the three control objectives as given earlier. According to this theorem the specification $R(s)$ should be replaced by a set of three control objectives, namely:

$$\Theta = \{[B_a, \{a\}], [B_b, \{b\}], [B_c, \{c\}]\}$$

We consider the derivation of condition B_a:

$$\begin{array}{ll} & B_a(s) \\ = & R(s) \ \wedge \ \neg R(sa) \\ = & (s \in \mathbf{pref}\{a^n b c^n\}) \ \wedge \ (sa \notin \mathbf{pref}\{a^n b c^n\}) \\ = & \quad [\text{ with } m \neq n \text{ and } m \geq 0\ ] \\ & (s \in \mathbf{pref}\{a^n b c^n\}) \ \wedge \ (s = a^n b c^m) \\ = & \underbrace{(s \in \mathbf{pref}\{a^n b c^n\})}_{1} \ \wedge \ \underbrace{(s \ \mathbf{N} \ b = 1)}_{2}\end{array}$$

leading to the control objective $[s \ \mathbf{N} \ b = 1, \{a\}]$. Notice that the first conjunction is always true and is not included.
It is left to the reader to verify that the other objectives are given by:

$$[s \ \mathbf{N} \ b = 1 \ \vee \ s \ \mathbf{N} \ a = s \ \mathbf{N} \ c, \{b\}]$$
$$[s \ \mathbf{N} \ b = 0 \ \vee \ s \ \mathbf{N} \ a = s \ \mathbf{N} \ c, \{c\}]$$

4 Unobservable events

Suppose the alphabet $\mathbf{a}P$ of P contains two kinds of events: observable and unobservable ones, denoted by $\mathbf{o}P$ and $\mathbf{u}P$ respectively. Control requirements usually are defined on the whole behaviour of P, i.e., on $\mathbf{b}P$. Because not all events are seen by the observer, the evaluation of the control objectives should depend on the observed behaviour.

Theorem 2 *If* $\mathbf{a}P = \mathbf{o}P \cup \mathbf{u}P$, *a set of control objectives* $\Theta = \{[B, e]\}$ *on the whole behaviour* $\mathbf{b}P$ *can be replaced by the set of observable control objectives*

$$\mathbf{oco}(\Theta) = \{[\mathbf{oco}(B), e]\}$$

on the observed behaviour $\mathbf{b}P\lceil\mathbf{o}P$, *where*

$$\mathbf{oco}(B)(t) = (\exists s : s\lceil\mathbf{o}P = t \ \wedge \ B(s))$$

However, $\mathbf{oco}(B)$ is only useful, if it is independent of any unobservable event. Therefore we introduce:

Definition 3 *The observable control objective* $\mathbf{oco}(B)$ *is decidable, if*

$$(\forall s\lceil\mathbf{o}P = t : \mathbf{oco}(B)(t) = B(s))$$

Property 2 *If* $\mathbf{oco}(B)$ *is decidable, we have*

$$B(s) = \mathbf{oco}(B)(s\lceil \mathbf{o}P)$$

If $\mathbf{oco}(B)$ is decidable then:

$$\begin{array}{ll} & (sbu)\,\mathbf{sat}[B, A] \\ = & B(s) \ \wedge \ b \notin A \\ = & \mathbf{oco}(B)(s\lceil \mathbf{o}P) \ \wedge \ b \notin A \\ \stackrel{def}{=} & (sbu)\,\mathbf{sat}[\mathbf{oco}(B), A] \end{array}$$

Therefore, we assume that systems with unobservable events have decidable observable control objectives. Otherwise, we can apply the previous theorem, but we do not get a computable result.

If in the set Θ a not-needed control objective is present, it will not affect $\mathbf{oco}(\Theta)$ as is shown by the following proposition:

Property 3

$$\begin{array}{l} (\forall s \in \mathbf{b}P : \neg B(s)) \\ \Rightarrow \\ (\forall s \in \mathbf{b}P\lceil \mathbf{o}P : \neg \mathbf{oco}(B)(s)) \end{array}$$

proof: directly from property 2.

If all $\mathbf{oco}(B)$ are decidable, we get the same controlled behaviour as if we would have been able to observe all events:

Property 4

$$\mathbf{b}P_{\mathbf{oco}(\Theta)} = \mathbf{b}P_{\Theta}$$

proof: if $\Theta = [B, a]$ then:

$$\begin{array}{ll} & \mathbf{b}P_{\mathbf{oco}(\Theta)} \\ = & \{t : t \in \mathbf{b}P \ \wedge \ t\,\mathbf{sat}\,\mathbf{oco}(\Theta)\} \\ = & \{t : t \in \mathbf{b}P \ \wedge \ t\,\mathbf{sat}[\mathbf{oco}(B), a]\} \\ = & \quad [\text{ let } t = sbu \text{ }] \\ & \{t : t \in \mathbf{b}P \ \wedge \ \mathbf{oco}(B)(s\lceil \mathbf{o}P) \ \wedge \ b \neq a\} \\ = & [\text{ } \mathbf{oco}(B) \text{ is decidable }] \\ & \{t : t \in \mathbf{b}P \ \wedge \ B(s) \ \wedge \ b \neq a\} \\ = & \{t : t \in \mathbf{b}P \ \wedge \ t\,\mathbf{sat}[B, a]\} \\ = & \{t : t \in \mathbf{b}P \ \wedge \ t\,\mathbf{sat}\,\Theta\} \\ = & \mathbf{b}P_{\Theta} \end{array}$$

Example 5 Recall the system P from example 1. Suppose the specification now is

$$R(s) = (s \in \mathbf{pref}\{a^*bc^*\})$$

According to the previous section the following set of control objectives are derived:

$$\Theta = \{[s\ \mathbf{N}\ b = 0, c], [s\ \mathbf{N}\ b = 1, \{a, b\}]\}$$

Suppose that event c is unobservable. According to theorem 2 we compute the set of observable control objectives as follows:

$$\begin{array}{ll} & \mathbf{oco}(\Theta) \\ = & [\{\mathbf{oco}(s\ \mathbf{N}\ b = 0), c], [\mathbf{oco}(s\ \mathbf{N}\ b = 1), \{a, b\}]\} \\ = & [\{t\ \mathbf{N}\ b = 0, c], [t\ \mathbf{N}\ b = 1, \{a, b\}]\} \end{array}$$

Notice, that in the second line we have $s \in \mathbf{b}P$, while in the last line we have $t \in \mathbf{b}P\lceil \mathbf{o}P$.
This example illustrates that we can specify desired behaviours which include unobservable events, and block these events, without observing them. If a is unobservable we are still able to get the desired behaviour! However, if b is unobservable the control objectives become undecidable and we cannot derive it.

5 Uncontrollable events

An event can only be blocked if it is controllable. Some events occur spontaneously and cannot be blocked. Such events are called uncontrollable. In general, the blocking event set in a control objective does not contain controllable only events. In these terms, the main problem to solve is to transform control objectives with uncontrollable events into control objectives where only controllable events are present. Given a system P, we know the set of controllable events, denoted by $\mathbf{c}P$ and the set of uncontrollable events $\mathbf{e}P$, sometimes called exogenous events, see [5].
A control objective in which the blocking set contains only controllable events is called a *controllable control objective* (CCO). An *uncontrollable control objective* (UCO) is defined by $[B, a]$ where $a \in \mathbf{e}P$.

The only way to derive a CCO from an UCO is to back-propagate the blocking of the uncontrollable event in the plant, i.e., instead of blocking the desired uncontrollable event, one or more controllable events that occur earlier in the behaviour of the plant should be blocked, so the uncontrollable event cannot occur anymore.
For example, if the only possible behaviour is abc and the control objective $[B, c]$ results in $B(ab) = \mathbf{true}$ then c should be blocked. If c is uncontrollable, we should block earlier, i.e., block b, if b is controllable. For this, we cannot use the same condition B, but should change it accordingly to B', such that $B'(a) = B(ab)$ returns true and b is blocked.

Definition 4 *For a set of control objectives $\Theta(P) = [B, a]$ we define the set of one step back-propagated control objectives by*

$$\mathbf{bco}(\Theta) = \{[\mathbf{bco}_e(B), e] : e \in \mathbf{before}(a)\}$$

where the before-set of event a is given by

$$\begin{array}{ll} & \mathbf{before}(a) \\ = & \{a' : a' \in \mathbf{a}P \ \wedge \ (\exists t, u \in (\mathbf{a}P)^* : ta'au \in \mathbf{b}P)\} \\ = & \ \cup \mathbf{init}(a) \end{array}$$

with

$$\mathbf{init}(a) = \begin{cases} \{\epsilon\} & \text{if } a \in \mathbf{b}P \\ \emptyset & \text{otherwise} \end{cases}$$

$\mathbf{before}(a)$ *is the set of events that occur just before the occurrence of the event a and it includes the trigger event in the case the system can start from the event a,*
and the back-propagated condition is

$$\mathbf{bco}_e(B)(s) = \begin{cases} B(se) & \text{if } e \in \mathbf{dom}(B) \\ B(s) & \text{if } e \notin \mathbf{dom}(B) \end{cases}$$

where $\mathbf{dom}(B)$ is the set of events from which strings are made of in order to evaluate the condition B, i.e., normally $\mathbf{dom}(B) = \mathbf{a}P$ but $\mathbf{dom}(\mathbf{oco}(B)) = \mathbf{o}P$.

Remark that $\mathbf{bco}_e(B)(S) = B(s)$ if s is not in the domain of the function B, i.e, if we do not use events like e to evaluate the conditions.

Notice that, because we assume $\mathbf{b}P$ to be prefix closed, the string u in the definition of $\mathbf{before}(a)$ can be omitted. Also notice that the "event" ϵ, as introduced here, corresponds to an unlabelled arrow pointing the initial state of a state graph.[1] The set $\mathbf{before}(a)$ can be computed easily in the case P is regular and is represented by a finite state automaton. The following informally described algorithm can then be used:

1. construct the reverse graph of the automaton, i.e., reverse the directions of all arrows.
2. compute, in the reversed automaton, all events that can occur after the occurrence of the event a. This can be done in some recursive way and is a finite process because the automaton contains only finite number of states and finite number of transitions.
3. add the trigger event ϵ in case the original system starts from the event a.
4. the computed set of events is the **before**-set of the original graph.

[1] ϵ as event is the trigger event, ϵ also can denote a string: the empty string. This evident double meaning should not lead to confusion.

If we use finitely recursive processes to model the system, the before-set is again derived from the model. In this case the desired behaviour may not be represented by a finite automaton. Since the representation and the number of events are finite the calculation of the set is again a finite process.

Next we prove that back-propagating control objectives does not lead to undesired behaviour:

Property 6

$$\mathbf{b}P_{\mathbf{bco}(\Theta)} \subseteq \mathbf{b}P_\Theta$$

proof: We give the proof only in the case the **before**-set does not contain the trigger event ϵ. If so, an additional condition should be taken into account which only complicates the expressions, but does not contribute to the proof. If $\Theta = [B, a]$ then:

$$\begin{array}{ll} & \mathbf{b}P_{\mathbf{bco}(\Theta)} \\ = & \{t : t \in \mathbf{b}P \ \wedge \ t \,\mathbf{sat}\, \mathbf{bco}(\Theta)\} \\ = & \{t : t \in \mathbf{b}P \ \wedge \ (\forall e \in \mathbf{before}(a) : t \,\mathbf{sat}\, [\mathbf{bco}_e B, e])\} \\ = & \quad [\text{ Consider } t = sbu \text{ }] \\ & \{t : t \in \mathbf{b}P \wedge \\ & \quad (\forall e \in \mathbf{before}(a) : \mathbf{bco}_e B(s) \ \wedge \ b \neq e)\} \\ = & \{t : t \in \mathbf{b}P \wedge \\ & \quad (\forall e \in \mathbf{before}(a) : B(se) \ \wedge \ b \neq e)\} \\ \subseteq & \quad [\text{ let } s' = se \text{ }] \\ & \{t : t \in \mathbf{b}P \ \wedge \ b \in \mathbf{a}P \ \wedge \ B(s') \Rightarrow b \neq a\} \\ = & \quad [\ t = s'bu \ ] \\ & \{t : t \in \mathbf{b}P \ \wedge \ t \,\mathbf{sat}\, \Theta\} \\ = & \ \mathbf{b}P_\Theta \end{array}$$

As stated before, some control objectives are not needed and do not contribute to the controlled system when blocking conditions never return true. The following property shows, that these control objectives, after back-propagation, still do not contribute and are not needed.

Property 7

$$(\forall s : \neg B(s)) \Rightarrow (\forall s, e : \neg \mathbf{bco}_e(B)(s))$$

proof: Directly from definition 4.

When an uncontrollable objective exists, we proceed as follows: we compute the back-propagated control objectives for the uncontrollable events, and repeat back-propagating until all objectives become controllable. However, proceeding in this way may lead to infinite repetitions, because first, $\mathbf{before}(a)$ may

contain a again, and second, before the occurrence of an event a in the behaviour of P an uncontrollable loop (loop with uncontrollable events) may be present.

The first case is technical and we can deal with it as described in [4]. However, when an uncontrollable loop exists the back-propagated procedure is finite only when the model is represented by a finite automaton where this loop can be identified. Then the loop can be avoided as it is described in [4]. When no finite state representation is available (for example the system is "dynamic" and is modelled by a finitely recursive process) the loop cannot be identified and therefore the back-propagation is not efficient.
Only if the existence of the loop is known a priori from the model, a computable result can be derived. However, this is consistent with the same assumption given in [6].

It is worth mentioning here that the complexity of the back-propagated algorithm depends on the number of the events of the model and on the complexity of the associated conditions.

6 Combination of results

Now suppose the alphabet of P contains both controllable/uncontrollable events and observable/unobservable events. In this case we can combine the results of the previous sections.

If $\Theta = \{[B_a, a]\}$ for an $a \in \mathbf{a}P$, then we have the following possibilities:

1. a is both controllable and observable:
   Θ need not be changed
2. a is controllable but is unobservable:
   Θ is replaced by: $\mathbf{oco}(\Theta) = \{\mathbf{oco}(B), a]\}$
3. a is uncontrollable but observable:
   Θ is replaced by: $\mathbf{bco}(\Theta) = \{[\mathbf{bco}_e(B), e] : e \in \mathbf{before}(a)\}$
4. a is both uncontrollable and unobservable:
   There are two ways to proceed: we may first deal with the unobservability or first deal with the uncontrollability. First dealing with unobservability leads to:

   $$\mathbf{bco}(\mathbf{oco}(\Theta)) = \{[\mathbf{bco}_e(\mathbf{oco}(B)), e] : e \in \mathbf{before}(a)\}$$

   while first dealing with uncontrollability leads to:

   $$\mathbf{oco}(\mathbf{bco}(\Theta)) = \{[\mathbf{oco}(\mathbf{bco}_e(B)), e] : e \in \mathbf{before}(a)\}$$

   The following property shows that both ways lead to the same set of control objectives (if all observable control objectives are decidable):

Property 8 *For decidable control objectives:*

$$\mathbf{oco}(\mathbf{bco}_e(B)) = \mathbf{bco}_e(\mathbf{oco}(B))$$

proof: First suppose $e \in \mathbf{o}P$, then:

$\mathbf{bco}_e(\mathbf{oco}(B))(s\lceil\mathbf{o}P)$
$= \mathbf{oco}(B)(s\lceil\mathbf{o}Pe)$
$= B(se)$
$= \mathbf{bco}_e(B)(s)$
$= \mathbf{oco}(\mathbf{bco}_e(B))(s\lceil\mathbf{o}P)$

Next, suppose $e \notin \mathbf{o}P$, then $e \notin \mathbf{dom}(\mathbf{oco}(B))$ and:

$\mathbf{bco}_e(\mathbf{oco}(B))(s\lceil\mathbf{o}P)$
$= \mathbf{oco}(B)(s\lceil\mathbf{o}P)$
$= B(s)$
$= \mathbf{bco}_e(B)(s)$
$= \mathbf{oco}(\mathbf{bco}_e(B))(s\lceil\mathbf{o}P)$

This last property says that we can solve the control problem using separation of concerns.

7 Conclusions

The control of a DES is introduced based on specifications involving predicates and blocking events. Algorithmic derivation involves partial observations and uncontrollability and it is shown that, under this formalism, these two issues can be formulated independently and solved separately.

References

[1] P.J. Ramadge and W.M. Wonham, "Supervisory control of a class of discrete event processes", *SIAM journal on Control and optimalisation*, vol. 25 (1), 1987, See also: systems control group report 8515, department of electrical engeneering, University of Toronto.

[2] P.J. Ramadge and W.M. Wonham, "The control of discrete event systems", *Proceedings of the IEEE*, vol. 77, 1989.

[3] R. Smedinga, *An Overview of Results in Discrete Event Systems using a Trace Theory Based Setting*, vol. 13 of *Progress in Systems and Control Theory*, pp. 43–56, Birkhäuser Verlag, Basel, Switzerland, 1993, (Proceedings of the Joint Workshop on Discrete Event Systems (WODES'92), Aug. 26–28, 1992, Prague, Czechoslovakia).

[4] M.P.Spathopoulos, R. Smedinga, and M.A. de Ridder., "Distributive control of logical discrete event systems using control objectives.", in A. Isidori, S. Bittanti, E. Mosca, A. De Luca, M.D. Di Benedetto, and G. Oriolo, editors, *Proceedings of the 3rd European Control Conference, Roma, Italy, september 1995*, 1995.

[5] R. Smedinga, *Control of discrete events*, PhD thesis, University of Groningen, 1989.

[6] S. Chung, S. Lafortune, and F. Lin, "Supervisory control using variable lookahead policies", *Discrete Event Dynamic Systems: Theory and Applications*, pp. 237–268, 1994.